PROMPTNESS DOES NOT IMPLY SUPERLOW CUPPABILITY

DAVID DIAMONDSTONE

Abstract. A classical theorem in computability is that every promptly simple set can be cupped in the Turing degrees to some complete set by a low c.e. set. A related question due to A. Nies is whether every promptly simple set can be cupped by a superlow c.e. set, i.e. one whose Turing jump is truth-table reducible to the halting problem $\emptyset'$. A negative answer to this question is provided by giving an explicit construction of a promptly simple set that is not superlow cuppable. This problem relates to effective randomness and various lowness notions.

1. Introduction

The notion of lowness is a central idea in computability theory, which intuitively expresses the idea that a set is computationally weak. It appears throughout computability theory and related fields. Significant examples include the low basis theorem of Jockush and Soare, as well as results in model theory, reverse mathematics, and effective randomness. Many of these theorems share a common theme, which is that low sets resemble computable sets, or else show that if one cannot find a computable example of a given object, one can at least find one which is low. Because being low is such a central idea which captures a natural property, it has generalizations, strengthenings, weakenings, and analogs in many other structures. One such fruitful analog is the idea of lowness in the theory of randomness. A surprising result here is that many natural lowness properties in randomness coincide and define the same class, known either as the K-trivials, or by other names which express the other equivalent lowness properties. (This combines several results by various researchers. A good survey paper is [4].)

One particular strengthening of the condition that a set A is low ($A' \leq_T \emptyset'$) is the requirement that there is actually a truth-table reduction from its jump to the halting problem ($A' \leq_{tt} \emptyset'$). If a set A satisfies this property, it is said to be superlow. This is natural, because it indicates that not only can one approximate the jump (as one can for any low set), but one can actually bound the number of mistakes the approximation makes before settling down to the correct value. It is also connected to important notions in randomness such as K-triviality: in [6], Nies proves that all K-trivials are superlow. Superlow sets also occur in studying notions of traceability, especially jump-traceability, which is another type of lowness

The author would like to thank Andreé Nies for originally asking the question and for generously sharing early drafts of his upcoming book, which were very helpful; the referee for corrections and suggestions, and for helping to streamline the proof; and Denis Hirschfeldt and Peter Cholak for reading early drafts and offering their advice. Great thanks are also due the author's advisor, Robert Soare, for suggesting the problem, for countless hours spent listening to attempts at finding a solution, and for his unwavering confidence that a solution would be found.

property which has appeared in connection with randomness; jump-traceability and superlowness coincide on the c.e. degrees.

When considering the computably enumerable sets, one can distinguish between static properties of sets and dynamic properties. A computably enumerable set A is enumerated in some order as a union of finite sets $\{A_n\}_{n\in\omega}$, and so one may define dynamic properties by considering not just the final set A produced, but also how it was produced; a static property is one which depends only on A, whereas a dynamic property depends on both A and on how it is enumerated in a particular enumeration. Lowness and superlowness are examples of static properties, because they depend only on the features of the set A, and not how it was produced. By contrast, an important dynamic property is that of prompt simplicity. A c.e. set B is promptly simple if there is an enumeration $\{B_n\}_{n\in\omega}$ and a computable "promptness" function $p : \omega \to \omega$ such that for all $e \in \omega$, if the eth c.e. set W_e is infinite, then there is some stage s and an element x which is enumerated into W_e at stage s, and appears in B by stage $p(s) \geq s$. (We say that W_e intersects B "promptly", hence the name.) In a beautiful paper [1], Ambos-Spies, Jockush, Shore, and Soare show that if a c.e. set B is promptly simple, then it is low cuppable: there is a low c.e. set A which cups B in the c.e. Turing degrees, i.e. $\emptyset' \leq_T A \oplus B$. In fact, they go even further, and show that a c.e. set is low cuppable exactly when it has the degree of a promptly simple set (such sets are called prompt, and have an alternate characterization in terms of permitting). This shows that static and dynamic properties can sometimes be equivalent; the static property of being low cuppable is equivalent to the dynamic property of being prompt. The promptly simple sets are then computationally strong, just as the low sets are weak: a c.e. set is of the same degree as a promptly simple set exactly when, with the help of some (computationally weak) low set, it can compute the halting problem.

It is natural to ask whether other notions of lowness share this property. In [2], Downey, Greenberg, Miller, and Weber show that the answer is yes for another lowness property, that of being array computable. In their main result, they show that every promptly simple set is cuppable to $\mathbf{0}'$ by an array computable c.e. set. In fact, their main result says more: the class of c.e. sets that can be cupped to $\mathbf{0}'$ by an array computable c.e. set properly contains the prompt sets. In contrast, the K-trivials do not share this property: there is an incomplete c.e. set which bounds the K-trivials (in fact, this set can be made low_2 [7]), but every upper cone in the c.e. sets (other than $\{\mathbf{0}'\}$) contains an incomplete promptly simple set, so there is an incomplete promptly simple set bounding the K-trivials. André Nies asked the corresponding question for superlowness: is every promptly simple set cupped to $\mathbf{0}'$ by some superlow c.e. set? This question attracted the attention of other researchers; see e.g. [2]. In this paper, we answer Nies's question.

Theorem 3.1. *There is a promptly simple set which is not superlow cuppable.*

One should note here that since the result of this paper was announced, André Nies used techniques from effective randomness to prove that all c.e. sets possessing another lowness property, that of being strongly jump traceable, are almost superdeep, which means that their join with any superlow set is superlow. In particular, they are not superlow cuppable. Since it was known that there is a promptly simple set which is strongly jump traceable, this gives a second proof of the main result of this paper. (For details, see [7].) However, the direct construction provides several advantages, such as different intuition and increased flexibility. The proof

in this paper provides a strategy to meet requirements preventing a set from being superlow cuppable, meaning that it can be combined with other constructions or modified in ways that the randomness theoretic proof cannot.

1.1. **Structure.** We begin by giving the intuition behind the solution of the problem by going through potential strategies for either direction, building up to an overview of the one that eventually works. This is done in section 2. We then actually prove the theorem by constructing a promptly simple set which is not superlow cuppable. The proof itself is in four parts. First we set up the requirements (3.1) that we will actually satisfy to make a set which is prompt but not superlow cuppable. The next section, 3.2, outlines the strategy used to satisfy a single negative requirement. The final two sections complete the proof: section 3.3 gives the construction itself; and section 3.4 performs the final verification.

2. Intuition

In trying to understand this problem, the first thing to do is recall the proof that every promptly simple set is low cuppable. Can it be modified to give us a superlow set which cups? Does it, perhaps, already give us a superlow cupping set? That proof is based on a simple and elegant idea. We are given a promptly simple set B, and to make a low set A which cups with B, we build a reduction Θ and put something new into A whenever we need to keep track of a change in $\emptyset'$. This looks like it would prevent A from being made low, but we can preserve computations $\Phi_e^A(e)$ to make A low by asking B to let us move the use of $\Theta^{A\oplus B}(i)$, which we imagine as a movable marker $\theta(i)$, past the use of $\Phi_e^A(e)$, which we imagine as a marker $\varphi(e)$, "clearing" the computation. In this way we can keep track of changes in $\emptyset'$ by making corresponding changes to A, but without making A nonlow. Since B is promptly simple, it must eventually grant these requests.

2.1. **A two player game.** We imagine this as a two-player game, in the tradition of Lachlan. In the low-cupping theorem, the blue player plays a low c.e. set A against the red player's promptly simple set B, in an attempt to make the join of A and B complete. The proof that promptly simple sets are *low* cuppable is viewed as a winning strategy for Blue, taking advantage of the fact that the prompt simplicity of B ensures that Red must eventually grant Blue's requests that B change in order to clear computations. The question now is who has a winning strategy if Blue is required to produce a *superlow* set. A key point in the low cupping theorem is that Red's promptly simple request-granting is only eventual. Red can fail to respond to Blue's requests as many times as he wants before finally acceding, and this poses a serious problem when we consider the superlow case.

2.2. Blue**'s difficulty in making** A **superlow.** Let us examine now the superlow case. The initial hurdle for Blue in making A superlow is that Red can wait before clearing a computation, potentially until after Blue's approximation to A' has exceeded some computable bound. This is the first hint that the superlow case may in fact go quite differently from the low case. When Blue is trying to make the cupping set superlow, he has a counter running down that keeps track of the number of changes he is allowed to make in some approximation to the jump. Ideally, Red can just wait until this counter runs out, and only then enumerate something into B to make it promptly simple. However, even though Blue has a counter, only he decides when he changes his approximation. The two players are

really engaged in a waiting game, and RED can't afford to wait forever, but must take positive action to make B promptly simple. If RED is to win, he must force BLUE to change his approximation to A'. However, BLUE controls A, so can always make any temporary computation $\Phi_e^A(e)[s]$ at some stage s disappear by changing A below the use. This looks hopeless for RED, and if we consider a single e in isolation it is. BLUE can keep changing A and never change his approximation to A' unless RED gives him a clear, as long as we are considering only a single e. RED's strategy must be to somehow exploit the interplay between different computations $\Phi_e^A(e)$ for different values of e.

2.3. **RED's strategy.** Suppose that RED controls all of the functionals Φ_e. This is not precisely true, but is true enough, since any threat RED might want to make is reflected somewhere in those functionals. RED can play some computation, say $\Phi_{10}^A(10)$, with large use. However, since BLUE can always wait for a clear, we assume he eventually gets one, and the use marker $\varphi(10)$ is cleared of all use markers $\theta(i)$ for BLUE's computation except for those few he is willing to let injure $\Phi_{10}^A(10)$. At this point, BLUE says, "Sure, $\Phi_{10}^A(10)$ converges," and changes his approximation accordingly. But here is RED's carefully laid trap: RED then plays a new computation, say $\Phi_0^A(0)$, with the same use as $\Phi_{10}^A(10)$. This new computation is clear of all computations that BLUE was willing to let injure the old computation $\Phi_{10}^A(10)$, but the new computation is more important, so BLUE is unwilling to allow the same level of injury. This puts BLUE in a bind, because either he needs to change his approximation to $\Phi_0^A(0)$ even though the use has *not* been cleared of the necessary computations (a bad idea), or else he must issue a new request to RED to clear this new computation. If RED fails to grant this request, then in order to keep issuing new requests BLUE must change A, intentionally injuring his own approximation to whether $\Phi_{10}^A(10)$ converges. This essentially allows RED to continue attacking computations which BLUE thought were cleared, and is the key to the RED strategy.

3. The proof

Theorem 3.1. *There is a promptly simple set B which is not superlow cuppable.*

Proof. We will construct such a set B.

3.1. **Setting up the proof.**
We may build B as a subset of the even numbers, and assume that prompt simplicity is measured relative to subsets of the even numbers (in the sense of equation 1 below), since any such B is equal to $\emptyset \oplus \hat{B}$ for some set $\hat{B}$ which is promptly simple in the usual way.[1] We'll make use of the fact that for any X, the condition $X \leq_{tt} \emptyset'$ is equivalent to the following condition:

- X is ω-c.e., i.e. there is a uniformly computable sequence of functions g_s which approximates X:

$$x \in X \iff \lim_{s\to\infty} g_s(x) = 1$$

[1]This allows us to consider c.e. sets A which are subsets of the odds, and take as our oracle $A \cup B$ instead of $A \oplus B$. The advantage of this formulation is that it allows us to directly compare the uses of functionals with oracle A and with oracle $A \cup B$, which simplifies matters later.

and a computable function h which bounds the number of changes made by the approximation g:

$$\text{for all } x,\ |\{s : g_{s+1}(x) \neq g_s(x)\}| \leq h(x)$$

Thus A is superlow if and only if A' is ω-c.e. When dealing with an enumeration of possible candidates for such a sequence g_s witnessing that A is superlow, we would, at first glance, have to deal with partial as well as total functions. However, this can be avoided by means of a trick.

Lemma 3.2. *Let g_s be a uniformly partial computable sequence of functions. Then we can obtain (uniformly from an index for g_s) a uniformly computable sequence $\hat{g}_s$ such that for all x, if g is total and $\lim_s g_s(x)$ exists, then $\lim_s \hat{g}_s(x) = \lim_s g_s(x)$. Moreover, the number of changes made by $\hat{g}$ is no greater than that made by g:*

$$\textit{for all } x,\ |\{s : \hat{g}_{s+1}(x) \neq \hat{g}_s(x)\}| \leq |\{s : g_{s+1}(x) \neq g_s(x)\}| + 1$$

Proof. Let $\hat{g}_s$ be defined as follows:

$$\hat{g}_s(x) = \begin{cases} g_u(x) & \text{where } u \text{ is the greatest } t \leq s \text{ such that } g_{t,s}(x)\downarrow \\ 0 & \text{if no such } u \text{ exists} \end{cases}$$

Fix x. Suppose $\lim_s g_s(x) = i$. Then there is some t so that $g_s(x) = i$ for all $s \geq t$. Let n be such that $g_{t,n}(x)\downarrow$. Now for $s > \max(t, n)$, we have $\hat{g}_s(x) = g_u(x)$ for some $u \geq t$, and hence $\hat{g}_s(x) = i$. Thus $\lim_s \hat{g}_s(x) = i$ as well.

For all s, let u_s be the u in the definition of $\hat{g}_s(x)$, or -1 in the case that there was no such u. Then the sequence u_s is monotonically increasing. For any s with $\hat{g}_{s+1}(x) \neq \hat{g}_s(x)$, therefore, either $u_s = -1$, or $g_{u_{s+1}}(x) \neq g_{u_s}(x)$, so there is some t, with $u_s \leq t < u_{s+1}$, such that $g_{t+1}(x) \neq g_t(x)$, and which is unique to s. Therefore we have

$$|\{s : \hat{g}_{s+1}(x) \neq \hat{g}_s(x)\}| \leq |\{s : g_{s+1}(x) \neq g_s(x)\}| + 1$$

as claimed. □

To ensure that B is not superlow cuppable, we need to meet, for each collection A, Θ, g_s, h (where A is a c.e. subset of the odd numbers, Θ is a Turing functional, g_s is a uniformly computable sequence of $\{0,1\}$-valued functions (made total using the trick from lemma 3.2), and h is (partial) computable) the condition

$$\text{If } \emptyset' = \Theta^{A\cup B} \text{ and } \lim_s g_s = A', \quad \text{then for some } x, \text{ either } h(x)\uparrow \text{ or } |\{s : g_s(x) \neq g_{s-1}(x)\}| > h(x).$$

If any of these conditions are not fulfilled, that produces an A with $\emptyset' \leq_T A \cup B$ and moreover the set A' is ω-c.e. Since any c.e. set is computable in $\emptyset'$, we can use this fact to show that $A \cup B$ can compute any c.e. set C. Similarly, $J^A = \lambda e.\Phi_e^A(e)$ is universal for partial-A-computable functions, so if $A' = \text{dom } J^A$ is ω-c.e., so is the domain of any A-computable partial function. Conversely, if we can build some c.e. set C and some Turing functional Ψ satisfying the requirement below, then B will not be superlow cuppable. Let $(A_k, \Theta_k, g_s^k, h_k)$ be a uniform listing of all possible collections A, Θ, g_s, h. It suffices to satisfy, for all k, the requirement

$$R_k:\ \text{If } C = \Theta_k^{A_k\cup B} \text{ and } \lim_s g_s^k = \text{dom } \Psi^{A_k}, \quad \text{then for some } x, \text{ either } h_k(x)\uparrow \text{ or } |\{s : g_{s+1}^k(x) \neq g_s^k(x)\}| > h_k(x)$$

We also need to make B promptly simple; that is, to ensure that $2\omega \setminus B$ is infinite and satisfy, for all e, the positive requirement

$$P_e : \text{ If } |V_e| = \infty, \text{ then } (\exists s)(\exists x)\, x \in V_{e,\text{ at } s} \cap B_s \tag{1}$$

where V_e is an effective listing of all c.e. subsets of the evens, and $V_{e,\text{ at } s}$ denotes the difference $V_{e,s} \setminus V_{e,s-1}$. This ensures that B is promptly simple via the identity function.

Our strategy for building the set B will be just as one would expect: we will place some restraints on B, and if at some stage s a new element enters V_e, the requirement P_e is not yet satisfied, and the element is not restrained by a higher priority restraint, we immediately place it in B. The whole construction is therefore in designing the restraints, and in building the set C and the functional Ψ.

3.2. **Meeting a single requirement R_k.** For convenience, we will drop the subscript k when no confusion will result. To meet requirement R_k, we produce a sequence $e_0 < e_1 < e_2 < \cdots$ with the intention that for some e_n, the number of times the approximation g_s changes its mind on e_n is greater than $h(e_n)$. This may fail to happen, but only if g_s fails to approximate $\operatorname{dom} \Psi^A$ or the functional $\Theta^{A\cup B}$ fails to compute C. The result will be that R_k is satisfied because either $C \neq \Theta^{A\cup B}$, $\operatorname{dom} \Psi^A \neq \lim_s g_s$, or h does not bound the number of "mind-changes" made by g_s.

The current stage will always be denoted s. We assume that at all stages, $\Theta^{A_s\cup B_s}$ is correctly computing C_s on the places of interest (which will be a set D defined below), and $g_s(e) = 0$ whenever $\Psi^A(e)\uparrow\ [s]$. Otherwise we can wait for one or the other to catch up.

The procedure to defeat the given A, Θ, g_s, h corresponding to R_k is the following:

(1) Pick a follower e_0, and wait for $h(e_0)\downarrow$. When it does, pick a set $D \subset \omega$ of size $h(e_0) + 1$. Set $n = 0$.

(2) Pick a follower e_{n+1}, wait for $h(e_{n+1})\downarrow$.

(3) Put $u = \max\{\theta(x) : x \in D\}$, and enumerate $(e_{n+1}, A_s \upharpoonright u)$ into the graph of Ψ. Now $\Psi(e_{n+1})\downarrow$ with use u. Wait for $g_s(e_{n+1}) = 1$. If, before this happens, either $A \upharpoonright u$ or $B \upharpoonright u$ changes, do the following:
   (a) If $A \upharpoonright u$ changes, but not $B \upharpoonright u$, return to the beginning of step 3.
   (b) If $B \upharpoonright u$ changes, cancel e_{n+1} and go to step 2.
   When $g_s(e_{n+1}) = 1$, set $m = n$, then increase n by 1.

(4) Enumerate $(e_m, A_s \upharpoonright u)$ into the graph of Ψ, so $\Psi(e_m)\downarrow$ with the same use u as in step 3. Impose B-restraint of priority $m+k$ and wait for $g_s(e_m) = 1$. If, before this happens, either $A \upharpoonright u$ or $B \upharpoonright u$ changes, do the following:
   (a) If $A \upharpoonright u$ changes, but not $B \upharpoonright u$, increase n by 1 and go to step 2.
   (b) If $B \upharpoonright u$ changes, cancel $e_m, e_{m+1}, \ldots, e_n$. Set $n = m - 1$. If $m > 0$, go to step 2; if $m = 0$, cancel D and go to step 1.
   When $g_s(e_m) = 1$, if $m > 0$, decrease m by 1 and go to the beginning of this step. If $m = 0$, continue to the next step.

(5) Enumerate an element of $D \setminus C$ into C. Increase n by 1 and go to step 2.

In addition to following the steps outlined above, we need to ensure that $\Theta^{A_s\cup B_s}$ is correctly computing C_s on D, and $g_s(e) = 0$ whenever $\Psi^{A_s}(e)\uparrow\ [s]$ on any of our followers e. So, whenever $A \upharpoonright u$ changes and so $\Psi^{A_s}(e)$ becomes undefined on

one of our followers, we wait for $g_s(e) = 0$ before carrying out further instructions. Similarly, whenever $A \upharpoonright u$, $B \upharpoonright u$, or $C \upharpoonright D$ changes, we wait for $\Theta^{A_s \cup B_s} \upharpoonright D = C_s \upharpoonright D$ before carrying out further instructions. In particular, this means that there is a hidden wait instruction in step 5 because once we change C, we have to wait for $\Theta^{A \cup B}$ to change in response. Note that the steps of this procedure are not presumed to take place during a single stage of the construction. There is an important point to make: "waiting" does not mean halting all action while waiting for (e.g.) $h(e)$ to converge (which may never happen). When the procedure gives a "wait" instruction, it is telling the construction not to perform any further actions *as part of this procedure* until the given condition is met, but to continue to perform actions in procedures for other requirements which may be running in parallel, and to continue to enumerate unrestrained elements into B, whenever instructed to do so, in order to make B promptly simple.

We can immediately check that this construction is well-defined. Firstly, every time it gives an instruction to choose a follower/agitator, the previous one (if any) will have been canceled. Secondly, any time some $\Psi(e)\downarrow$ for a follower e used in the procedure, the use will be the same $u = \max\{\theta(x) : x \in D\}$, and an instruction will be given to redefine $\Psi(e)$ only after an A change below u allows us to do so, or we have cancelled the old e and chosen a new one. Additionally, we point out that since the only B-restraint imposed has priority $m+k$, the procedure for R_k imposes restraint only of priority k and lesser priorities. Finally, we have the following:

Lemma 3.3. *If the procedure for R_k is permanently stuck in a wait step, it succeeds and R_k is satisfied.*

Proof. The only wait steps are to wait for some $h(e)$ to converge, to wait for $g_s(e) = 0$ (respectively 1) when $\Psi(e)\uparrow$ (respectively, $\Psi(e)\downarrow$), and to wait for $\Theta^{A_s \cup B_s} \upharpoonright D = C_s \upharpoonright D$. So if the procedure is permanently stuck in a wait step, it succeeds because either h fails to be total, $\lim_s g_s$ fails to compute $\mathrm{dom}\Psi$, or Θ fails to compute C. □

This leaves two possibilities: either the procedure will at some point reach its goal and be stopped by the construction, or else it will continue to take actions forever. The latter is possible, but will result in only finite restraint of any given priority, and R_k will still be satisfied in this case as $\Theta^{A \cup B}$ will not be total.

3.3. **Construction.**

Stage $s = 0$: Set $B_0 = \emptyset$, $C_0 = \emptyset$, graph $\Psi = \emptyset$.

Stage $s > 0$: Begin the procedure for R_{s-1}.

For any k (which will necessarily be at most $s-1$) and any follower e being used by the procedure for R_k, if the amount of changes made by the approximation g^k thus far exceeds the bound:

$$|\{t < s : g^k_{t+1}(e) \neq g^k_t(e)\}| > h_k(e)$$

immediately halt the procedure for R_k. Otherwise continue each procedure currently in progress one step further.

If a new element $x > 4e$ enters V_e, the requirement P_e is not yet satisfied, and x is not restrained by a restraint of priority $p \leq e$, enumerate x into B_s.

This ends the construction.

3.4. **Verification.**

Lemma 3.4. *There are finitely many restraints of any given priority p imposed over the course of the construction.*

Proof. Since the strategy for requirement R_k imposes restraint of priority k or of lesser priority, it suffices to show that each given strategy imposes restraint of each priority at most finitely often. Suppose instead that the strategy for R_k imposes priority p restraint infinitely often. (Note: $p \geq k$.) The strategy for R_k must impose priority p restraint infinitely often after some stage when each requirement P_j for $j < p + 1$ which ever acts has already acted, since each acts at most once.

Priority p restraint is imposed every time step 4 is performed, with $m = p - k$. Each time this happens, there is a change to $g^k_s(e_m)$, or else there is a change to $g^k_s(e_{m+1})$. There cannot have been a B change, because B is restrained and we assumed that each positive requirement capable of violating the restraint has already acted or will never act. So each time restraint is imposed there is a mind-change by g on either e_m or e_{m+1}. Moreover, though e_m or e_{m+1} can be cancelled during the construction, they cannot be at this point because they are only cancelled when a B change violates restraint. So there is some e (either e_m or e_{m+1}) such that the change set

$$\{t : g^k_{t+1}(e) \neq g^k_t(e)\}$$

is infinite. But then there is some stage s by which its size has exceeded the bound:

$$|\{t < s : g^k_{t+1}(e^k_n) \neq g^k_t(e^k_n)\}| > h_k(e^k_n)$$

(we guaranteed that $h_k(e)$ converged in step 1 or 2), and so the procedure for R_k is halted, and no more restraint is ever imposed for requirement k, giving the desired contradiction. □

Corollary 3.5. *The set B constructed is promptly simple.*

Proof. Fix e. The total amount of restraint of priority $p \leq e$ is finite. If V_e is infinite, then eventually some element larger than all priority $p \leq e$ restraint enters V_e, and then immediately afterwards enters B. So B is promptly simple via the identity function. (We know that $\overline{B} \cap 2\omega$ is infinite since requirement P_e puts at most one element x into B, and only if $x > 4e$, so B contains at most e of the first $2e$ even numbers.) □

Lemma 3.6. *The set B is not superlow cuppable.*

Proof. Suppose that B is superlow cuppable. Then for some k the requirement R_k is violated. Let us examine the strategy for R_k. By lemma 3.3, if the procedure for R_k is permanently stuck in a wait step, then R_k is satisfied. We may therefore assume that all steps in the procedure halt. Furthermore, since the procedure itself only halts when R_k is satisfied, we may assume that the procedure never halts, and so infinitely often carries out, to completion, at least one of the steps 1–5. Step 1 is carried out only when the procedure is first called or after a B change violates a restraint of priority k, so in particular at most finitely often.

Any time step 2 is carried out it is immediately followed by step 3, and any time step 4 is carried out it is immediately followed by step 1 or step 2, so it must be the case that step 3 is carried out infinitely often after the step 1 is carried out the final time. Each time step 3 is carried out, either there is an A or B change below $u = \max\{\theta(x) : x \in D\}$, or the procedure goes to step 4. Each time step 4 is

carried out, there is either an A or B change below u, or the procedure goes to step 5. Finally, each time step 5 is carried out, there must be an A or B change below u before any further steps are taken, since C changes, and the only way $\Theta^{A\cup B}$ and C can be brought back into agreement is an A or B change below u. So every time step 3 is carried out, there is an A or B change below $\max\{\theta(x) : x \in D\}$ before it is carried out again, so there are infinitely many changes below $\max\{\theta(x) : x \in D\}$. (Note that D only changes when step 1 is carried out, so the D here is always the same.) This is only possible if one of these θ markers moves infinitely often, so R_k is satisfied because Θ is not total. □

References

[1] K. Ambos-Spies, C. G. Jockusch, Jr., R. A. Shore and R. I. Soare, An algebraic decomposition of the recursively enumerable degrees and the coincidence of several degree classes with the promptly simple degrees, *Transaction of the American Mathematics Society*, vol. 281 (1984), pp. 109–128.

[2] R. Downey, N. Greenberg, J. Miller, and R. Weber, Prompt simplicity, array computability and cupping, in Chong, Feng, Slaman, Woodin, and Yang (eds.), *Computational Prospects of Infinity*, Lecture Notes Series, Institute for Mathematical Sciences, National University of Singapore, vol. 15, World Scientific (2008), pp. 59–78.

[3] R. Downey, N. Greenberg, and R. Weber, Totally ω-computably enumerable degrees I: bounding critical triples, *Journal of Mathematical Logic* vol. 7 (2007), pp. 145–171.

[4] R. Downey, D. R. Hirschfeldt, A. Nies, and S. A. Terwijn, Calibrating Randomness, *Bulletin of Symbolic Logic*, vol. 12 (2006), pp. 411–491.

[5] A. Kučera and T. A. Slaman, Low upper bounds of ideals, to appear.

[6] André Nies, Lowness properties and randomness, *Advances in Mathenatics*, vol. 197 (2005), pp. 274–305.

[7] André Nies, *Computability and Randomness*, Clarendon Press, Oxford, to appear.

[8] Robert I. Soare, *Recursively Enumerable Sets and Degrees*, Springer-Verlag, Heidelberg, 1987.

[9] Robert I. Soare, *Computability Theory and Applications*, Springer-Verlag, Heidelberg, to appear.

University of Chicago, 5734 S. University Avenue, Chicago, Illinois 60637
E-mail address: ded@math.uchicago.edu